NASA Contractor Report 191463

ICASE Report No. 93-23

ICASE 20 Years of Excellence

OPTIMAL CUBE-CONNECTED CUBE MULTIPROCESSORS

Xian-He Sun
Jie Wu

NASA Contract No. NAS1-19480
May 1993

Institute for Computer Applications in Science and Engineering
NASA Langley Research Center
Hampton, Virginia 23681-0001

Operated by the Universities Space Research Association

National Aeronautics and
Space Administration
Langley Research Center
Hampton, Virginia 23681-0001

Optimal Cube-Connected Cube Multicomputers [*]

Xian-He Sun

ICASE
NASA Langley Research Center
Hampton, VA 23681-0001

Jie Wu

Department of CSE
Florida Atlantic University
Boca Raton, FL 33431

Abstract

Many CFD (computational fluid dynamics) and other scientific applications can be partitioned into subproblems. However, in general the partitioned subproblems are very large. They demand high performance computing power themselves, and the solutions of the subproblems have to be combined at each time step. In this paper, the cube-connect cube (CCCube) architecture is studied. The CCCube architecture is an extended hypercube structure with each node represented as a cube. It requires fewer physical links between nodes than the hypercube, and provides the same communication support as the hypercube does on many applications. The reduced physical links can be used to enhance the bandwidth of the remanding links and, therefore, enhance the overall performance. The concept and the method to obtain optimal CCCubes, which are the CCCubes with a minimum number of links under a given total number of nodes, are proposed. The superiority of optimal CCCubes over standard hypercubes has also been shown in terms of the link usage in the embedding of a binomial tree. A useful computation structure based on a semi-binomial tree for divide-and-conquer type of parallel algorithms has been identified. We have shown that this structure can be implemented in optimal CCCubes without performance degradation compared with regular hypercubes. The result presented in this paper should provide a useful approach to design of scientific parallel computers.

[*]This research was supported in part by the National Aeronautics and Space Administration under NASA contract NAS1-19480 while the first author was in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA 23681-0001.

1 Introduction

Rapidly advancing technology has made it possible for a large number of processors to be interconnected to form a single multiprocessor system. In recent years, the multiprocessor approach has been shown to be the most straightforward and cost-effective way for achieving high performance. However, the way in which processors, memory modules, and switches should be interconnected to form an efficient architecture remains a research issue. Parallel computers have been built with a variety of architectures. One of the popular parallel architectures is the hypercube architecture [16], also known as the binary n-cube, which contains 2^n processors, each of which is connected by fixed communication links to n other nodes. The value n is known as the dimension of the hypercube. In a hypercube structure two nodes are connected if and only if their addresses differ in one and only one bit.

The hypercube structure has many desirable properties. It is symmetric. Any n dimensional cube can be divided into two $n-1$ dimensional cubes. Many other topologies, such as ring, mesh, and tree, can be mapped into the hypercube topology. It is rich in connection, a message can be transferred from one node to all the other nodes in a total of n steps in an n-cube. Extensive research efforts have been focused on hypercube design aspects and hypercube applications. Most of the first generation and second generation distributed-memory multiprocessors are based on hypercube architecture. Examples of these commercial products include FPS's T series, Ncube's nCUBE, Ametek's S/14, Intel's iPSC, and Thinking Machine's Connection Machine, which is a hypercube interconnected bit-serial SIMD machine.

Efforts have also been made to vary the hypercube topology to obtain better interconnection networks. Many variations of the hypercube topology, such as twisted hypercubes [5], enhanced hypercubes [21], extended hypercubes [11], bridged hypercubes [3], incomplete hypercubes [10] and Fibonacci cubes [7], balanced hypercubes [8] and folded hypercubes [4], etc., have been proposed. These new architectures keep the desirable properties of hypercubes, and incorporate new features that are more suitable for some specific applications and objectives. The Cube-Connected Cube (CCCube) structure [23] is one of the variations of hypercube topology. A CCCube is an extended hypercube structure with each node represented as a cube. With the same number of processors, A CCCube requires few physical links than a comparable hypercube and provides the same support as the hypercube does in many ways. The routing and broadcasting algorithms in the CCCube have been discussed in several previous studies [6], [23].

The parallel divide-and-conquer paradigm is a computation paradigm which partitions a single complex problem into a set of subproblems, which are further divided until every independent subproblem has been broken up sufficiently. After all the subproblems have been solved, data (or results) are collected. The above process can be represented by a *binomial tree* structure. Lo *et al.* [12] have shown that the binomial tree is an ideal computation structure for parallel divide-and-conquer algorithms, and is superior to the classic full binary tree structure with respect to speedup and efficiency. Since a large number of parallel algorithms are divided-and-conquer in nature, the ability to embed (or map) a binomial tree into a network can be considered an important measure of the network.

This paper studies the capability of embedding a binomial tree in a CCCube. We first prove that an i-level binomial tree can be embedded in any (m, n)-CCCube, where m is the dimension of the outer cube and n is the dimension of the inner cube, provided that $m + n \geq i$. With the objective of embedding a binomial tree in a CCCube using as few links as possible, we define an optimal CCCube as being one with the minimum number of links for a given number of processors. Reducing the number of links will lead to a higher bandwidth of the remanding links, lower network contention, and thus better overall performance. The selection of an optimal CCCube for a given binomial tree is also provided in this paper. Comparison is made between CCCubes and standard hypercubes in terms of the link usage in the embedding of binomial trees. We also identify a class of parallel algorithms that is best suited for optimal CCCube structures. This class of parallel algorithms is based on the semi-binomial tree proposed in this paper.

This paper is organized as follows: Section 2 discusses embedding binomial trees in CCCubes. The determination of optimal cube-connected cubes is discussed in Section 3. Section 4 identifies a class of parallel algorithms based on the semi-binomial tree structure. A parallel merge sorting example is used to illustrate how to run the proposed algorithm on optimal CCCubes. Section 5 presents conclusions. A comprehensive comparison of CCCubes with other cube-based systems has been done in [22], and a comparison of CCCubes with Cube-Connected Cycles (CCC) [15] can be found in [13]. The use of CCCubes in other applications can be found in [23] and [24].

2 Embedding of Binomial Trees in CCCubes

An (m, n) cube-connected cube [23], or (m, n)-CCCube, is defined as an m-dimensional hypercube (outer-cube) with each node in the hypercube being an n-dimensional hypercube (inner-cube). Assume that $g_m g_{m-1} ... g_1 l_n l_{n-1} ... l_1$ is the binary address associated with each of the 2^{m+n} nodes in an (m, n)-CCCube, where $g_m g_{m-1} ... g_1$ is the global address and $l_n l_{n-1} ... l_1$ is the local address. The least significant bit, g_1, of the global address will be referred as global dimension 1, and so on. Similarly, the least significant bit of the local address designates local dimension 1, and so on. There are m global dimensions and n local dimensions in an (m, n)-CCCube. More formally, we have the following recursive definition of an (m, n)-CCCube:

Definition 1 • *A $(0, n)$-CCCube is an n-dimensional hypercube Q_n, with one node in $(0, n)$-CCCube a designated* port *node.*

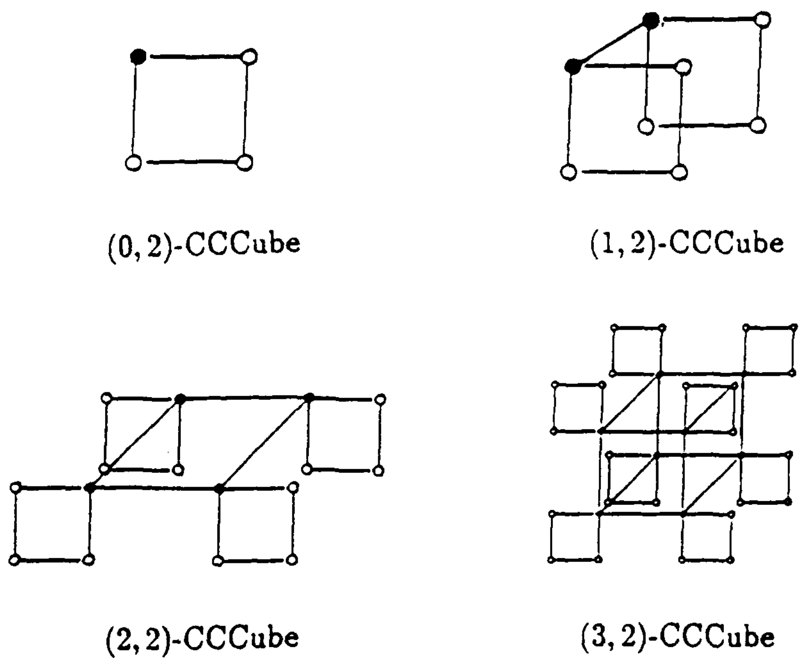

Figure 1. Constructions of (3, 2) CCCube

- *Suppose G and G' are disjoint $(m-1, n)$-CCCubes for $m \geq 1$. Then the graph obtained by adding edges between all the port nodes in G and the corresponding port nodes in G' is an (m, n)-CCCube. All the port nodes in G and G' are the port nodes in this (m, n)-CCCube.*

Figure 1 illustrates the rule for building a (3, 2)-CCCube. Basic prope^r^ties of a CCCube have been studied in [23], as well as routing and broadcasting algorithms.

The cube-connected cube architecture has many desirable properties. If we view the inner-cubes as nodes, then the outer-cube forms the hypercube architecture. Therefore, the architecture is symmetric, rich in connection, and can be partitioned into subcubes. The nodes in each inner-cube provide much higher computation power than a single processor. This two-level hypercube architecture fits many scientific applications well. For instance, the 3-D turbulence simulation code CDNS (Compressible Direct Simulation of Navier-Stokes) [17], which is used in and out of the NASA Langley Research Center for basic research in the physics of compressible homogeneous turbulence, calculates spatial derivatives with a sixth-order compact scheme. The compact scheme requires solutions of a large sparse system with multiple right sides, where each right side can be solved on an inner-cube concurrently, and then the solutions of each inner-cube can be combined through the outer-cubes in the next time step. In general, the two-level computation, or partition computational paradigm, is applicable to any simulation based on the compact scheme. It is also applicable to any simulation code based on the alternating direction implicit (ADI) method and the fast Poisson's solvers [17].

CCCubes also support any program paradigm supported by hypercubes. For example, the total data-exchange communication [19], the data-gathering communication, and the data-scattering communication [18] all requires $\log(n)$ communication steps on an n-dimension hypercube, therefore, they require no more than $\log(m) + \log(n)$ communication steps on a (m, n)-CCCube. In many cases, the CCCube architecture provides better support than a two-level hypercube. As we mentioned in Section 1, the divide-and-conquer paradigm is one of the dominating computation paradigms in parallel processing. The partition paradigm given above can be seen as a special case of the divide-and-conquer paradigm. In this section, we prove that CCCube provides hypercube-like support for the divide-and-conquer paradigm.

One of the most conventional graph representations of divide-and-conquer algorithms is the *binomial tree* [1]. More specifically, an i-level binomial tree, B_i, can be recursively defined as follows:

Definition 2

- *Any tree consisting of a single node is a B_0 tree.*
- *Suppose that T and T' are disjoint B_{i-1} trees, for $i \geq 1$. Then the tree obtained by adding an edge to make the root of T become the leftmost offspring of the root T' is a B_i tree.*

Figure 2 shows the construction of high level binomial trees from low level binomial trees. Lo *et al* [12] show the binomial tree structure as an ideal computation structure for parallel divide-and-conquer algorithms, and show its superiority to the classic full binary tree structure, with respect to speedup and efficiency. Therefore it is important to study the embedding of a binomial tree into a CCCube. In general, the embedding problem on cube-based systems [2], a restricted version of the mapping problem [14], is the problem of mapping a particular graph structure G to a cube-based system G'. The goal of the mapping problem is to find a mapping that minimizes the length of the path between communication processes in this graph structure G. Reducing the length of the communication path is important. Even with the new routing schemes, such as wormhole routing or circuit switching, shortening the path length will reduce the network contention and achieve better performance [20]. *Dilation* and *congestion* are two measures used to measure the quality of an embedding, where dilation is the maximum length in G' of the image of an edge of G and congestion of an edge of G' is the number of images of edges of G that pass through it.

Theorem 1 *An i-level binomial tree can be embedded with unit dilation in any (m, n) CCCube, provided that $m + n \geq i$. In addition, the root node of this i-level binomial tree can be mapped to any port node in the (m, n)-CCCube.*

Proof: We only need to show that an i-level binomial tree can be embedded in any (m, n) CCCube, where $m + n = i$. We prove it by using induction on m. When $m = 0$, any $(0, i)$-CCCube

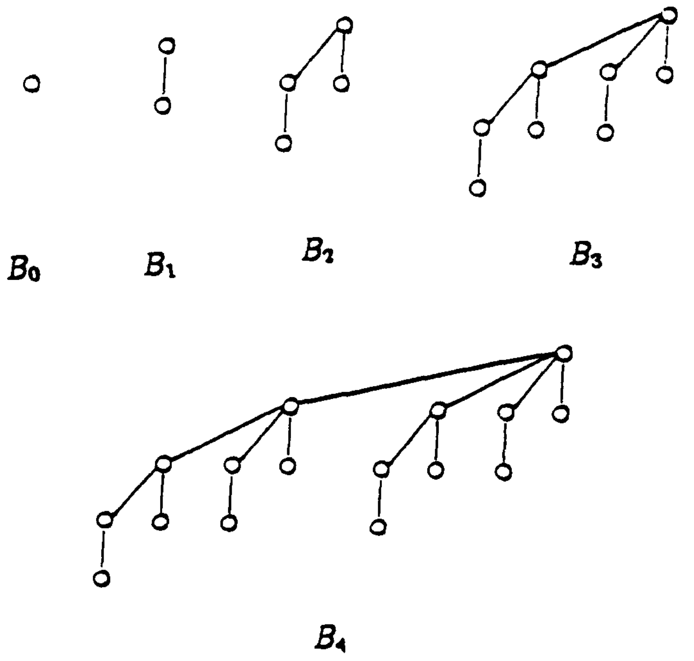

Figure 2. Binomial trees

is an i-dimensional hypercube Q_i. Therefore, an i-level binomial tree can be embedded in this $(0, n)$-CCCube [16] and the root node will be mapped to the only port node in the $(0, i)$-CCCube. Suppose when $m \leq i - 1$, a i-level binomial tree B_i, with $i \geq m$ can be embedded in any (m, n)-CCCube, such that $m + n = i$, and its root node to one of the port nodes. When $m = i$, a i-level binomial tree, B_i, with $i > m$,[1] can be decomposed into two disjoint $(i - 1)$-level binomial trees: B_{i-1} and B'_{i-1} with an edge connecting two root nodes of these two trees. Also, any (m, n)-CCCube can be decomposed into two $(m - 1, n)$-CCCubes, G and G', with edges connecting the port nodes of G and G'. Based on the assumption, B_{i-1} can be embedded in G with the root node assigned to any one of the port nodes, say a, in G. Similarly, B_{i-1} can be embedded in G' with the root node assigned to the port node a', the matching node of a in G. Since a and a' are connected in the (m, n)-CCCube, the edge that connects the root node of B_{i-1} and B'_{i-1} can be mapped to the edge that connects a and a'. $\square$

3 Finding the Optimal Cube-Connected Cube

Let m, n be the dimension of the outer-cube and the inner-cube, respectively. The following theorem determines how to choose m (or n) based on a constant $c = m + n$, i.e., a fixed number of nodes,

[1]We don't need to consider the case where $i = m$, since the corresponding $(i, 0)$-CCCube is an i-dimensional hypercube.

such that the (m,n)-CCCube has a minimum number of links.

Note that in an (m,n)-CCCube, the total number of nodes $|V| = 2^{m+n} = 2^c$ and the total number of links $|E| = c \cdot 2^{c-1} - \frac{m(2^c-2^m)}{2}$. We represent $c = 2^k + l$, where $0 \le l \le 2^{k-1}$, that is, $k = \lfloor \log c \rfloor$ and $l = c - 2^{\lfloor \log c \rfloor}$.

Theorem 2 *To obtain an (m,n)-CCCube with a minimum number of links, the selection of m, under a given constant $c = m+n$, where $c = 2^k + l, 0 \le l \le 2^k - 1$, is as follows:*

1. *If $l \ge k-2$ then $m = 2^k + l - k - 1$, namely $m = c - \lfloor \log c \rfloor - 1$, and the minimum number of links is $|E| = 2^{c-\lfloor \log c \rfloor -1}(c + (\lfloor \log c \rfloor + 1)2^{\lfloor \log c \rfloor +1} - \lfloor \log c \rfloor)$.*

2. *If $l \le k-2$ then $m = 2^k + l - k$ and the minimum number of links is $|E| = 2^{c-\lfloor \log c \rfloor}(c + \lfloor \log c \rfloor 2^{\lfloor \log c \rfloor} - \lfloor \log c \rfloor + 1)$.*

Proof: When $c = m+n$ is fixed, to obtain the minimum value $c \cdot 2^{c-1} - \frac{m(2^c-2^m)}{2}$ of the number of links in an (m,n)-CCCube, with a given constant $c = m+n$, is equivalent to obtaining the maximum value of $f(m) = m(2^c - 2^m)$. Note that $f(m+1) - f(m) = (m+1)(2^c - 2^{m+1}) - m(2^c - 2^m) = 2^c - 2^m(m+2)$ is monotone decreasing. Therefore, at $p = max\{m+1 | f(m+1) - f(m) \ge 0\}, f(m)$ reaches its maximum value, $f(p)$. Also, if $f(p) - f(p-1) = 0$, both $f(p)$ and $f(p-1)$ have the maximum value.

To find p we first determine its range by considering the following two cases:

1. If $m = 2^k + l - k + 1$, then

$$\begin{aligned} f(m+1) - f(m) &= 2^c - 2^{2^k+l-k+1}(2^k + l - k + 1 + 2) \\ &= 2^{c-k+1}(2^{k-1} - 2^k - l + k - 3) \\ &= -2^{c-k+1}(2^{k-1} + l + 3 - k) < 0; \end{aligned}$$

therefore, $p \le 2^k + l - k + 1$.

2. If $m = 2^k + l - k - 1$, then

$$\begin{aligned} f(m+1) - f(m) &= 2^c - 2^{2^k+l-k-1}(2^k + l - k - 1 + 2) \\ &= 2^{c-k-1}(2^{k+1} - 2^k - l + k - 1) \\ &= 2^{c-k-1}(2^k - 1 - l + k) > 0; \end{aligned}$$

therefore, $p \ge 2^k + l - k$.

Table 1. Optimal selection of m's under given c's, $1 \leq c \leq 32$

c, p		c, p		c, p		c, p	
1	0	9	6,7	17	14	25	21
2	1	10	7	18	14,15	26	22
3	2	11	8	19	15	27	23
4	2,3	12	9	20	16	28	24
5	3	13	10	21	17	29	25
6	4	14	11	22	18	30	26
7	5	15	12	23	19	31	27
8	6	16	13	24	20	32	28

Table 2. The number of links in optimal CCCubes and in compatible hypercubes

c, l_{cube}, $l_{occcube}$			c, l_{cube}, $l_{occcube}$		
1	1	1	9	960	2304
2	3	4	10	1984	5120
3	8	12	11	4096	11264
4	20	32	12	8448	24576
5	44	80	13	17408	53248
6	96	192	14	35840	114688
7	208	448	15	73728	245760
8	448	1024	16	151552	524288

With the above determined range of p, let us examine the case where $m = 2^k + l - k$,

$$\begin{aligned} f(m+1) - f(m) &= 2^c - 2^{2^k+l-k}(2^k + l - k + 2) \\ &= -2^{c-k}(l - k + 2) \end{aligned}$$

Therefore, when $l - k + 2 \leq 0$, $p = 2^k + l - k + 1$; and when $l - k + 2 \geq 0$, $p = 2^k + l - k$. □

Table 1 shows those p's under given c's, with c ranging from 1 to 32. Table 2 compares optimal CCCubes with compatible hypercubes in terms of number of links used, where c stands for the dimension of hypercubes, $l_{occcube}$ for the number of links in optimal CCCubes, and l_{cube} for the number of links in hypercubes. Figure 3 shows the optimal CCCube structure with c ranging from 1 to 5.

Figure 4 shows the comparison between the standard hypercube and the optimal CCCube in terms of link usage in the embedding of binomial trees, which is measured by the number of edges in a binomial tree divided by the total number of edges in hypercubes or optimal CCCubes.

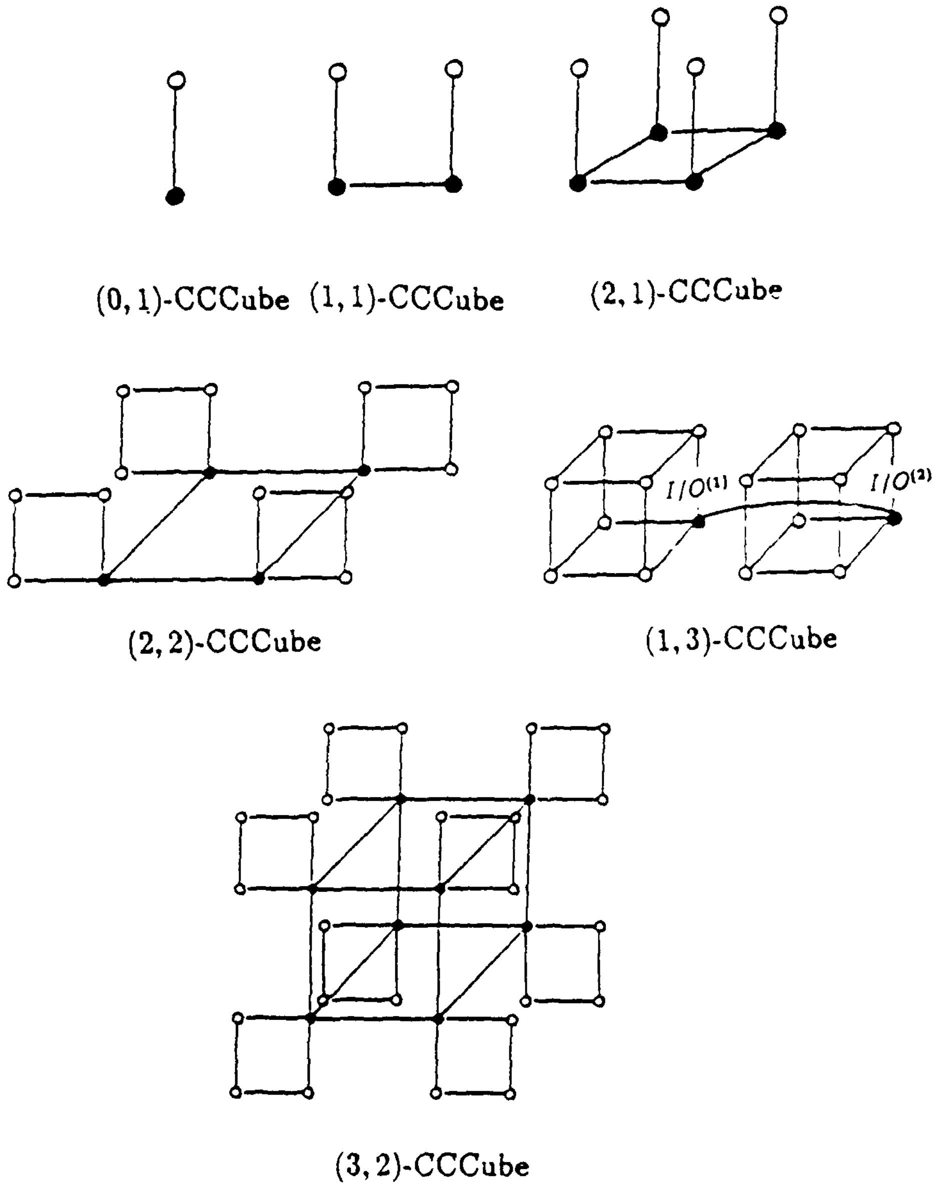

Figure 3. Optimal CCCubes

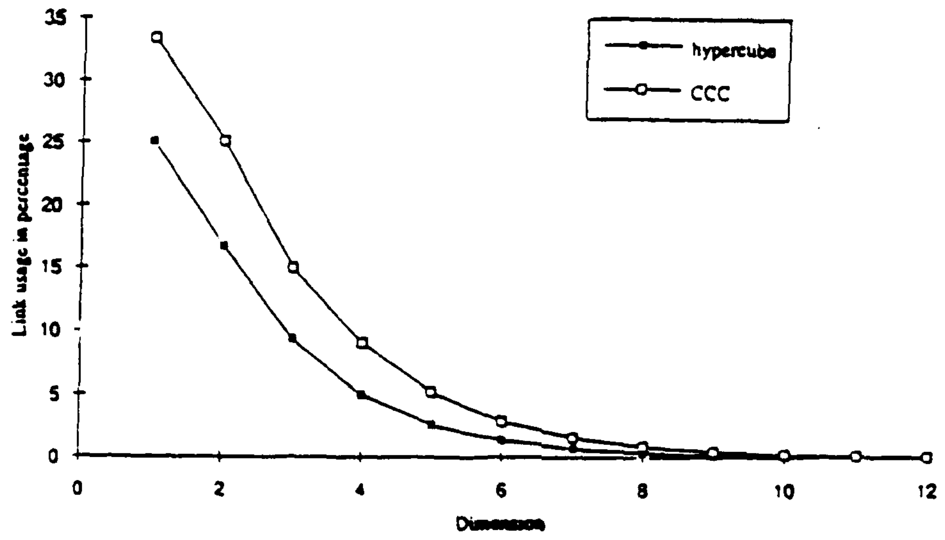

Figure 4. Link usage in standard hypercubes and optimal CCCubes

4 Execution of Parallel Algorithms on Optimal CCCubes

The most conventional graph representations of parallel-and-conquer algorithms are trees, such as binary trees and binomial trees. Divided-and-conquer algorithms normally involve three steps[9]: broadcasting, computation, and aggregation. The broadcasting phase distributes load to different nodes from one or more I/O nodes which has I/O function. The load should be evenly allocated to all nodes to reduce total execution time. The computation phase performs the computation required by each subproblem. The aggregation phase is normally a reverse procedure of broadcasting, and represents a collection process of results.

We study a computation structure based on a *semi-binomial tree* to implement parallel divide-and-conquer algorithms. In a semi-binomial tree, every node in the second level of the tree is the root node of a binomial tree. Figure 5 shows a semi-binomial tree with two second level nodes each of which is the root node of a B_3. In a CCCube structure, if we use the host as the root node of a semi-binomial tree and each I/O node (normally a port node) as the node at the second level of the tree, we can easily construct a spanning semi-binomial tree. For example, when both port nodes in the optimal (1,3)-CCCube are I/O nodes, the semi-binomial tree in Figure 5 is the corresponding spanning tree.

The outline of a parallel divide-and-conquer algorithms based on the semi-binomial tree structure is as follows:

1. Give the host the problem to be solved.

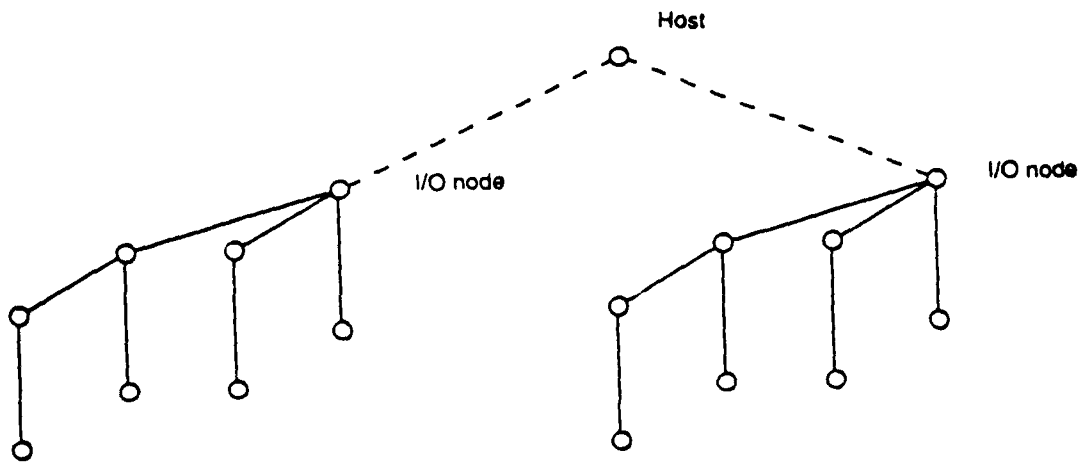

Figure 5. A semi-binomial tree

2. The host divides the problem into m subproblems and assigns each to a distinct I/O node in a CCCube. Normally m is the number of I/O nodes.

3. Each I/O node (the root node of a binomial tree) divides the subproblem in half and passes the first half to the child which has the most descendants and has not yet received work. The same process is applied to the second half, until all children receive work.

4. Every node performs the required work associated with each subproblem.

5. The results are passed back to each I/O node, and are merged in the reverse order when subproblems are passed down the tree.

6. The host collects results from each I/O node.

Note that in the above scheme, step 1 to step 3 corresponds to the broadcasting phase. Step 4 is the computation phase where every node computes at the same step. Steps 5 and 6 are the aggregation phase. To prevent potential bottleneck at the host, computations at step 1 and step 6 should be relatively light.

We use the merge sorting algorithm to illustrate the proposed approach. Suppose a list of 32 elements (3, 2, 12, 7, 5, 1, 13, 45, 23, 43, 8, 0, 11, 34, 15, 16, 4, 9, 25, 30, 21, 31, 54, 78, 89, 93, 63, 64, 29, 20, 10, 41) is to be sorted in the optimal (1, 3)-CCCube with two I/O nodes: $I/O^{(1)}$ and

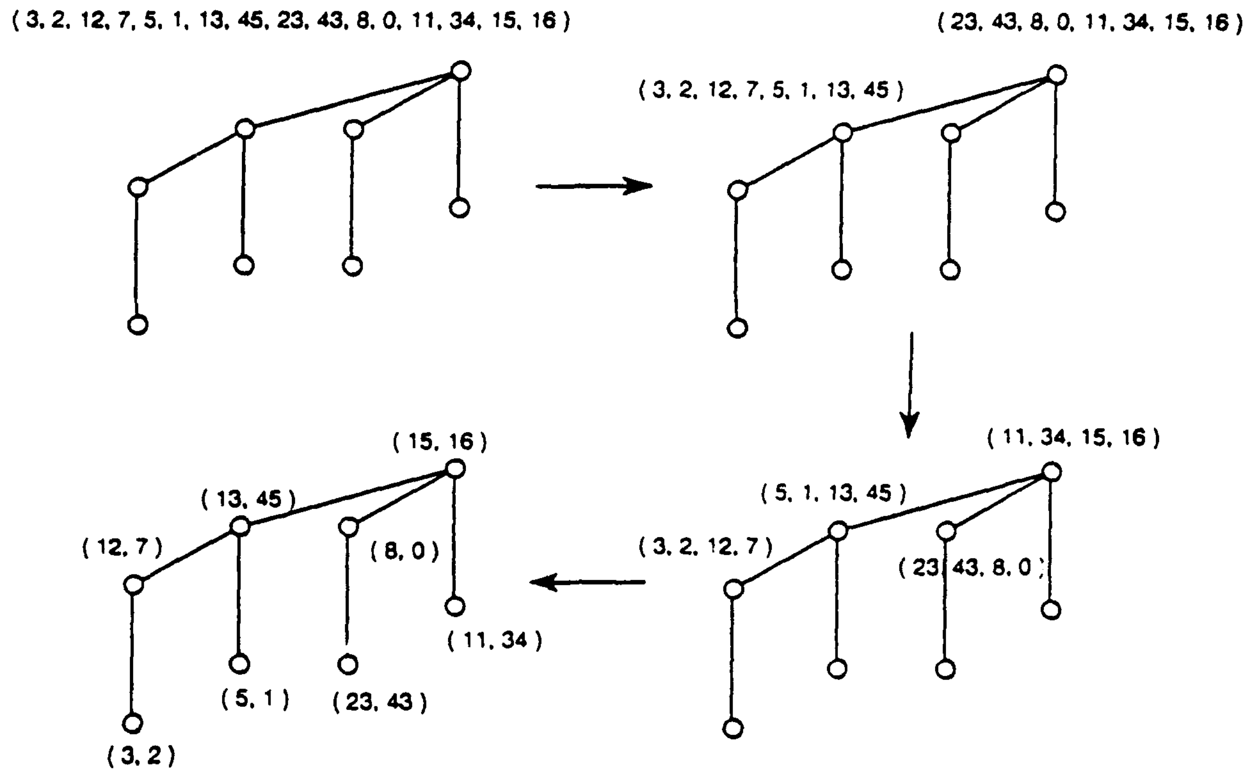

Figure 6. Broadcasting phase of merge sorting

$I/O^{(2)}$(see Figure 3). First, the host divides the list into two sublists of length 16. Suppose $I/O^{(1)}$ receives sublist (3, 2, 12, 7, 5, 1, 13, 45, 23, 43, 8, 0, 11, 34, 15, 16). The sorting process of a sublist assigned to $I/O^{(1)}$ is demonstrated in Figures 6 and 7. Figure 6 shows the broadcasting process. At the computation step every node, including $I/O^{(1)}$, performs a swap operation of two elements if necessary. The aggregation phase (Figure 7) resembles the broadcasting phase, but the message is distributed in the reverse order. At the end of the aggregation phase, the $I/O^{(1)}$ has the sorted sublist (0, 1, 2, 3, 5, 7, 8, 11, 12, 13, 15, 16, 23, 34, 43, 45). Similarly, $I/O^{(2)}$ has the sorted sublist (4, 9, 10, 20, 21, 25, 29, 30, 31, 41, 54, 63, 64, 78, 89, 93). Finally, the host collects and merges these two sorted sublists.

The proposed parallel divide-and-conquer algorithms can be implemented in regular CCCubes and hypercubes. Since there is no performance degradation when they are implemented in the CCCubes which use the fewest number of links, the optimal CCCube is a cost-effective structure for implementing this class of algorithms.

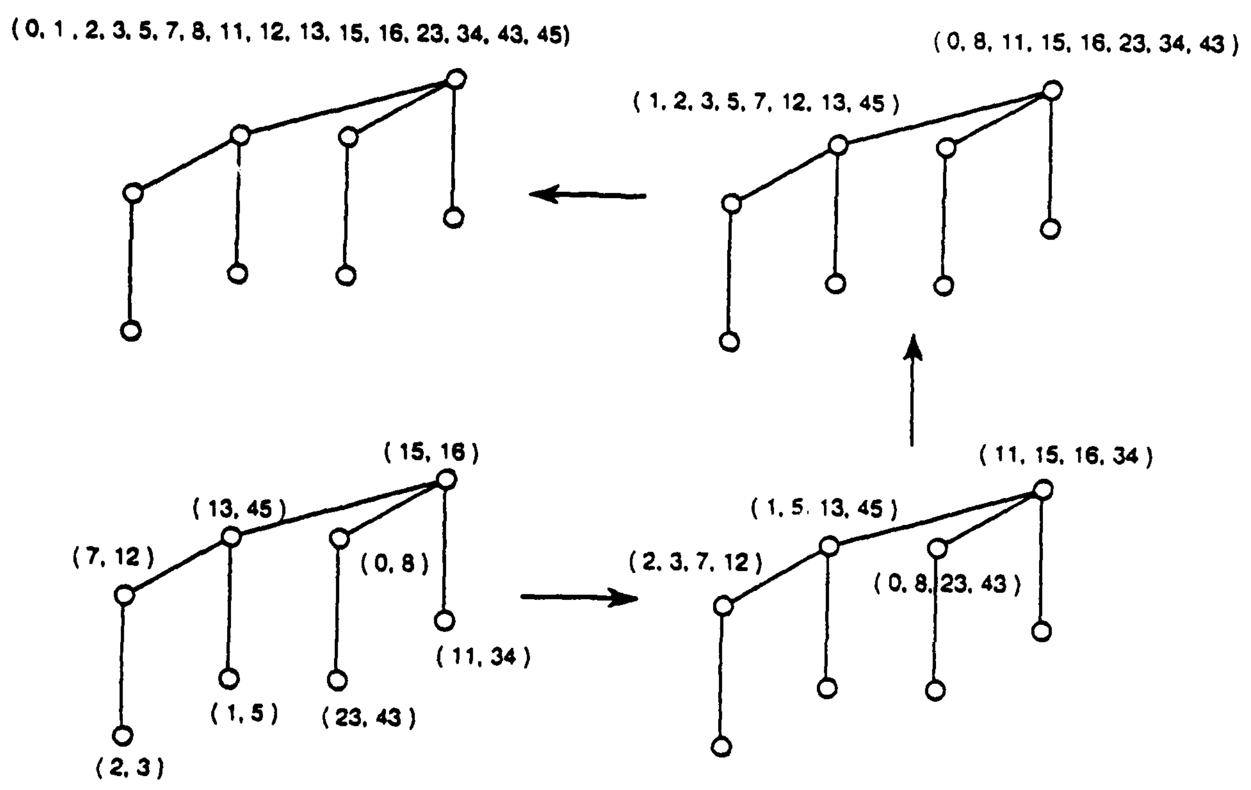

Figure 7. Aggregation phase of merge sorting

5 Conclusions

This paper explored in detail some properties of the Cube-Connected Cube (CCCube) structure, a variant of the hypercube structure, with each node replaced by a cube. We considered first the embedding of binomial tree, a useful structure for divide-and-conquer types of parallel algorithms, into a CCCube. It was proved that an i-level binomial tree can be embedded into any (m, n)-CCCube, where m is the dimension of outer cube and n is the dimension of the inner cube, provided that $m + n \geq i$. With the objective of embedding a binomial tree into a CCCube with a *minimum number of links*, the selection of an optimal (m, n)-CCCube under a given constant $c = m + n$ was provided in this paper. Comparison was also made between an (m, n)-CCCube with a c-dimensional hypercube in terms of the link usage in the embedding of a c-level binomial tree. A class of parallel divide-and-conquer algorithm was proposed based on a semi-binomial tree structure. It was shown that optimal CCCube is a cost-effective structure to implement such class of algorithms.

References

[1] BROWN, M. R. Implementation and analysis of binomial queue algorithms. *SIAM Journal of Computing*. Aug. 1978, 161-164.

[2] CHEN, W. K., STALLMANN, M., AND GEHRINGER, E. Hypercube embedding heuristics: An ecaluation. *International Journal of Parallel Programming*. 18, (6), 1989, 505-549.

[3] EL-AMAWY, A., AND LATIFI, S. Bridged hypercube networks. *Journal of Parallel and Distributed Computing*. 1990, 90-96.

[4] EL-AMAWY, A., AND LATIFI, S. Properties and performance of folded hypercubes. *IEEE Tran. on parallel and distributed systems*. 2, (1), Jan. 1991, 31-42.

[5] ESFAHANIAN, A., NI, L. M., AND SAGAN, B. E. The twisted n-cube with application to multiprocessing. *IEEE Transaction on Computers*. 40, (1), Jan. 1991, 88-93.

[6] GOYAL, P., AND FERNANDEZ, E. Cube-connected cubes - a recursively defined network architecture for parallel computation. *Proc. 4th Conf. on Hypercubes*. March 1989.

[7] HSU, W. J., PAGE, C. V., AND LIU, J. S. Computing prefixes on a large family of interconnection topologies. *Proceedings of the 1992 International Conference on Parallel Processing*. Vol 3, Aug. 1992, 153-159.

[8] HUANG, K., AND WU, J. Balanced hypercubes. *Proc. of the 1992 International Conference on Parallel Processing*. Vol 3, Aug. 1992, 80-84.

[9] JAMIESON, L. H., GANNON, D., AND DOUGLASS, R. J. *The Characteristics of Parallel Algorithms*. The MIT Press, 1987.

[10] KATSEFF, H. Incomplete hypercubes. *Hypercube Multiprocessors*. M. T. Heath, Ed., 1982, 258-264.

[11] KUMAR, J. M., AND PATNAIK, L. M. Extended hypercube: A hierarchical interconnection network of hypercubes. *IEEE Trans. on Parallel and Distributed Systems*. 3, (1), Jan. 1992, 45-57.

[12] LO, V. M., RAJOPADHYE, S., GUPTA, S., KELDSEN, D., MOHAMED, M. A., AND TELLE, J. Mapping divide-and-conquer algorithms to parallel architectures. *Proc. 1990 International Conference on Parallel Processing*. 1990, III, 128-135.

[13] LUO, Y., AND WU, J. Gray-code-based cube-connected cubes. to appear in Congressus Numerantium, 1993.

[14] NI, L., AND KING, C. T. On partition and mapping for hypercube computing. *International Journal of Parallel Programming*. 17, (6), 1988, 475-495.

[15] PREPARATA, F., AND VUILLEMIN, J. The cube-connected cycles, a versatile network for parallel computation. *Comm. of ACM*. May 1981, 30-39.

[16] SAAD, Y., AND SCHULTZ, M. H. Topological properties of hypercubes. *IEEE Transactions on Computers*. 37, (7), July 1988, 867-872.

[17] Sun, X.-H., and Joslin, R. A simple parallel prefix algorithm for compact finite-difference scheme. ICASE Technical Report, 93-16, ICASE, NASA Langley Research Center, 1993.

[18] Sun, X.-H., and Ni, L. A structured representation for parallel algorithm design on multicomputers. In *Proc. of the Sixth Conf. on Distributed Memory Computing* (April 1991).

[19] Sun, X.-H., Ni, L., Salam, F., and Guo, S. Compute-exchange computation for solving power flow problems: The model and application. In *Proc. of the Fourth SIAM Conf. on Parallel Processing for Scientific Computing* (Dec. 1989).

[20] Sun, X.-H., Zhang, H., and Ni, L. Efficient tridiagonal solvers on multicomputers. *IEEE Transactions on Computers 41*, 3 (1992), 286–296.

[21] Tzeng, N. F., and Wei, S. Enhanced hypercubes. *IEEE Trans. on Computers.* 40, (3), March 1991, 284-294.

[22] Wu, J. Broadcasting in injured hypercubes using limited global information. TR-CSE-92-39, Dept. of Computer Science and Engineering, Florida Atlantic University, Nov. 1992.

[23] Wu, J., and Larrendo-Petrie, M. Cube-connected-cube network. *Microprocessing and Microprogramming.* 33, (5), 1992, 299-310.

[24] Wu, J., and Wu, T. An efficient vector-matrix-vector multiplication on cube-connected-cubes multicomputers. to appear in International Journal of Mini and Microcomputers, 1993.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE May 1993	3. REPORT TYPE AND DATES COVERED Contractor Report

4. TITLE AND SUBTITLE OPTIMAL CUBE-CONNECTED CUBE MULTIPROCESSORS	5. FUNDING NUMBERS C NAS1-19480 WU 505-90-52-01
6. AUTHOR(S) Xian-He Sun Jie Wu	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Institute for Computer Applications in Science and Engineering NASA Langley Research Center Hampton, VA 23681-0001	8. PERFORMING ORGANIZATION REPORT NUMBER ICASE Report No. 93-23
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Langley Research Center Hampton, VA 23681-0001	10. SPONSORING/MONITORING AGENCY REPORT NUMBER NASA CR-191463 ICASE Report No. 93-23
11. SUPPLEMENTARY NOTES Langley Technical Monitor: Michael F. Card Final Report	Subm. to Int'l J. on Micro-computer Applications on Parallel & Multiprocessor Architectures
12a. DISTRIBUTION/AVAILABILITY STATEMENT Unclassified - Unlimited Subject Category 62	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Many CFD (computational fluid dynamics) and other scientific applications can be partitioned into subproblems. However, in general the partitioned subproblems are very large. They demand high performance computing power themselves, and the solutions of the subproblems have to be combined at each time step. In this paper, the cube-connect cube (CCCube) architecture is studied. The CCCube architecture is an extended hypercube structure with each node represented as a cube. It requires fewer physical links between nodes than the hypercube, and provides the same communication support as the hypercube does on many applications. The reduced physical links can be used to enhance the bandwidth of the remanding links and, therefore, enhance the overall performance. The concept and the method to obtain optimal CCCubes, which are the CCCubes with a minimum number of links under a given total number of nodes, are proposed. The superiority of optimal CCCubes over standard hypercubes has also been shown in terms of the link usage in the embedding of a binomial tree. A useful computation structure based on a semi-binomial tree for divide-and-conquer type of parallel algorithms has been identified. We have shown that this structure can be implemented in optimal CCCubes without performance degradation compared with regular hypercubes. The result presented in this paper should provide a useful approach to design of scientific parallel computers.

14. SUBJECT TERMS parallel processing, parallel architectures, hypercube, cube-connected cube, optimal cube-connected cube, divide-and-conquer paradigm, CFD applications	15. NUMBER OF PAGES 16
	16. PRICE CODE A03

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

☆U.S. GOVERNMENT PRINTING OFFICE: 1993 - 728-064/86005